A Neural Emulator for Uncertainty Estimation of Fire Propagation

Andrew Bolt[†], Conrad Sanderson[†‡], Joel Janek Dabrowski[†], Carolyn Huston[†], Petra Kuhnert[†◇]

[†] *Data61 / CSIRO, Australia;* [◇] *Australian National University, Australia;* [‡] *Griffith University, Australia*

Abstract

Wildfire propagation is a highly stochastic process where small changes in environmental conditions (such as wind speed and direction) can lead to large changes in observed behaviour. A traditional approach to quantify uncertainty in fire-front progression is to generate probability maps via ensembles of simulations. However, use of ensembles is typically computationally expensive, which can limit the scope of uncertainty analysis. To address this, we explore the use of a spatio-temporal neural-based modelling approach to directly estimate the likelihood of fire propagation given uncertainty in input parameters. The uncertainty is represented by deliberately perturbing the input weather forecast during model training. The computational load is concentrated in the model training process, which allows larger probability spaces to be explored during deployment. Empirical evaluations indicate that the proposed model achieves comparable fire boundaries to those produced by the traditional SPARK simulation platform, with an overall Jaccard index (similarity score) of 67.4% on a set of 35 simulated fires. When compared to a related neural model (emulator) which was employed to generate probability maps via ensembles of emulated fires, the proposed approach produces competitive Jaccard similarity scores while being approximately an order of magnitude faster.

1 Introduction

Wildfires are a destructive phenomenon affecting natural flora and fauna as well as posing many risks to human life and property. Response and management efforts are critical to mitigating the damage caused by wildfires [7, 8, 40]. To this end, in silico wildfire simulations are often performed in order to aid managers in making informed choices for fire risk management and mitigation efforts, as well as evacuation planning.

There are a number of computational fire spread models that are currently deployed for simulating and predicting the spreading behaviour of fires [17, 33, 45]. These models typically take into account underlying empirical observations of fire spread given specific conditions, such as fuel class (eg. vegetation type), terrain characteristics (eg. slope), as well as weather (eg. temperature, wind speed and direction). These simulations are usually deterministic [33, 45], and predictions depend on static initial conditions (initial fire location, fuel class, terrain maps) and estimates of future weather conditions.

As wildfire spread is inherently a stochastic process, a single simulation of fire progression is unlikely to account for *uncertainty* in the underlying variables driving the model. For example, there are variabilities within fuel classes, uncertainty in classifying a fuel, and uncertainty in weather forecasting. As such, it is desirable to produce likelihood estimates of fire location (probability map) at a given point in time, in order to provide a form of uncertainty quantification.

One approach for dealing with uncertainty in a given variable is to run an ensemble of simulations, where each simulation draws the value of the variable from a distribution. Ensemble predictions are a standard approach in areas such as weather and flood forecasting [19, 30, 48]. However, a major limitation of ensemble-based simulations is that the parameter space increases exponentially with the number of variables. The size of the ensembles is hence constrained by the computational demands of the simulations, which in turn can limit the scope of uncertainty analysis to a subset of variables.

To address this shortcoming, in this paper we explore the use of a spatio-temporal neural-based model to *directly* estimate the output of an ensemble of simulations. The proposed model is trained via probability maps, which are generated from ensembles of fire-fronts produced by the traditional SPARK simulation platform [33]. Within the training ensembles, weather parameters are perturbed according to a noise model. The computational load is hence concentrated in the model training process, rather than during deployment. Empirical evaluations show that the proposed model generalises well to various locations and weather conditions, and generates probability maps comparable to those *indirectly* generated via ensembles of SPARK simulations.

The paper is continued as follows. Section 2 provides a brief overview of related work. Section 3 describes the proposed model architecture. Section 4 provides comparative evaluations. We conclude by discussing the results, limitations and future work in Section 5.

* **Published in:** Procedia Computer Science, Vol. 222, pp. 367–376, 2023. DOI: 10.1016/j.procs.2023.08.175

2 Related Work

Recent work has investigated use of machine learning methods for wildfire related applications, covering topics such as susceptibility prediction, fire spread prediction, fire ignition detection and decision support models [2, 7, 24]. Deep learning architectures have also been used to develop models related to wildfires [4, 9, 22, 37]. Burge et al. [9] and Hodges et al. [22] present convolutional neural network (CNN) emulators that estimate fire spread behaviour, while Allaire et al. [4] present a CNN emulator for hazard assessment within a target region.

Neural networks have also been used to predict the likelihood of a region being burned, given the confidence the model has in its own prediction. Radke et al. [37] propose a CNN based emulator that predicts the likelihood a pixel is burned within a 24 hour period. Sung et al. [44] also develop a model that predicts the likelihood that a pixel is burned; the model is trained using daily fire perimeters, rather than synthetic data. Dabrowski et al. [12] combine a Bayesian physics informed neural network with level-set methods [31] for modelling fire-fronts. Outside of the fire space, Gunawardena et al. [20] deploy a hybrid linear/logistic regression based surrogate model to estimate the behaviour of ensemble weather modelling. Egele et al. [16] propose an automated approach for generating an ensemble of neural networks to aid uncertainty quantification.

In a separate line of research, surrogate models (also known as model emulators) can be used as more computationally efficient representations of complex physical process models [3, 39]. Such models are often based on statistical or machine learning approaches. Models of various physical systems have been achieved using techniques such as logistic regression [20], Gaussian processes [5, 11], random forests [18, 29] and deep neural networks [25, 26, 39, 42].

Potential advantages of neural network based model emulators include accelerated computation through the use of Graphics Processing Units (GPUs) via open-source packages such as TensorFlow [1] and PyTorch [36]. Furthermore, since neural network models are in essence a large set of numerical weight parameters, they are portable across computing architectures, easily deployed in cloud-based computing environments, and can be integrated into larger frameworks with relative ease [39].

3 Model Architecture

We extend the spatio-temporal neural network model architecture presented in [6]. As illustrated in Fig. 1, the architecture comprises an *Encoding* block, an *Inner* block, and a *Decoding* block. Compared to [6], the inner block is expanded with two layers: layer C implements edge detection, and layer G incorporates update constraints.

There are three types of inputs to the model: (i) initial probability map, (ii) spatial data and forcing terms, (iii) weather time series. The probability map consists of an image where pixels values represent the likelihood of the fire reaching a location at a given point in time. The spatial data are images representing height maps and land classes (eg. forest, grassland). The forcing terms are curing and drought factors. The weather time series consists of the mean and standard deviations of temperature, wind (x and y components), and relative humidity. These are summary statistics of the values used in the ensemble of simulations used for training.

The encoder and decoding steps are as described in [6]. The encoding step is used to create a more compact latent representation of features. This is illustrated by the light blue shading in Fig. 1. The probability map is encoded using an autoencoder, illustrated in Fig. 2. The relationship between pixel value and probability is preserved in the encoding process. The autoencoder component of the model is trained separately and its weights are frozen during the main training step. This transfer learning approach ensures that dynamics are not inadvertently learned by the encoding/decoding layers, as well as reducing the number of weights that need to be trained by the full model. The height map and landclass map features are encoded using alternating convolutional and strided convolutional layers which reduce the spatial extent.

The inner model is designed to handle the dynamics of the system, incorporating a shallow U-Net structure [38, 46]. See Fig. 3 for details. New latent probability map estimates are generated, which are then propagated forward in time as new weather values are processed. The updates are also informed by the latent spatial and climate features.

Within the inner model, layer C applies a Sobel edge filter [13], where the output is x and y edge components. The edge detection aims to make it easier for the neural network to identify where fire fronts are within the probability map. This approach has parallels with saliency guided training [23]. Fig. 4 shows examples of edge magnitudes.

Layer G implements constrained updates to the latent probability map. A new map ${P}^{t+1}$ at time $t+1$ is generated using the previous map ${P}^{t}$ and update terms from ${F}^{t}$ at time t, where F is the output layer from the U-Net component of the model (see Fig. 3). F uses a sigmoid activation function so that values lie on the $[0, 1]$ interval [10].

More specifically, layer G acts to update the latent probability map as follows. Let F_{ij}^{t} denote the value of the i-th kernel of F at pixel location j at time t. Let $P_j^{t} \in [0, 1]$ be the probability that pixel j has been burned by time t. The model is trained such that F_{ij}^{t} terms estimate the likelihood that fire spreads from a surrounding local region to location j at time $t+1$. The spread characteristics and likelihoods are learned by the model in the U-Net component. The probability that pixel j has been burned at time $t+1$ is estimated using:

$$P_j^{t+1} = G(P_j^{t}, F_j^{t}) = 1 - \left(1 - P_j^{t}\right) \prod_i \left(1 - F_{ij}^{t}\right) \tag{1}$$

In the above equation, the updated likelihood is given by 1 minus the probability that the pixel is not burned by an exterior source $\left(1 - F_{ij}^{\,t}\right)$, and is not already burned $\left(1 - P_j^{\,t}\right)$. The update satisfies the following three constraints:

1. The likelihood a pixel is burned is non-decreasing and bounded by 1, ie., $1 \geq P_j^{\,t+1} \geq P_j^{\,t}$.
2. The likelihood a pixel is burned is unchanged if there are no external sources, ie., $P_j^{\,t+1} \rightarrow P_j^{\,t}$ if $\forall\, i : F_{ij}^{\,t} \rightarrow 0$.
3. If there is at least one source that is very likely to burn a pixel, then the likelihood the pixel is burned approaches 1, ie., $P_j^{\,t+1} \rightarrow 1$ if $\exists\, i$ s.t. $F_{ij}^{\,t} \rightarrow 1$.

We have empirically observed that in order for the model to converge well in training, it is necessary to ensure that the F_i terms are sufficiently small so that the latent probability map is gradually updated between iterations. More specifically, preliminary experiments indicated that the model is unlikely to converge during training when using the default value of 0 for the initial bias of layer F. As such, we have empirically set the initial bias to -5.0.

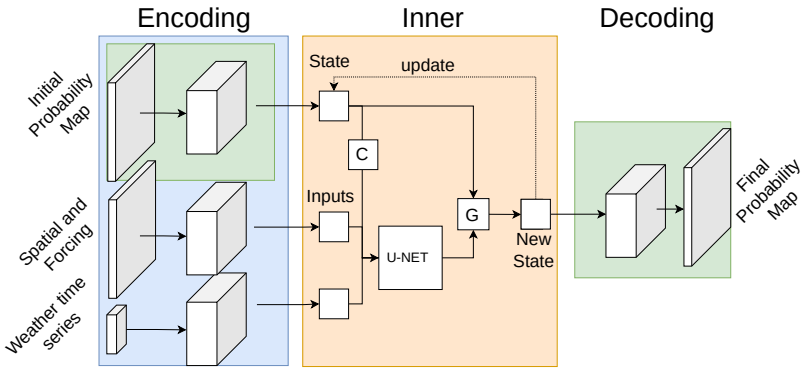


Figure 1: The proposed model architecture based on [6]. Key additions are in the inner component: edge detection (layer C) and constrained update layer (layer G). The initial probability map and features are encoded via down-sampling. The inner component of the model updates the latent probability map using spatial features and weather variables (see Fig. 3 for details). Once all weather inputs have been used a final latent probability map estimate is produced and decoded via up-sampling. The layers used to encode and decode probability maps are trained separately as an autoencoder (see Fig. 2 for details).

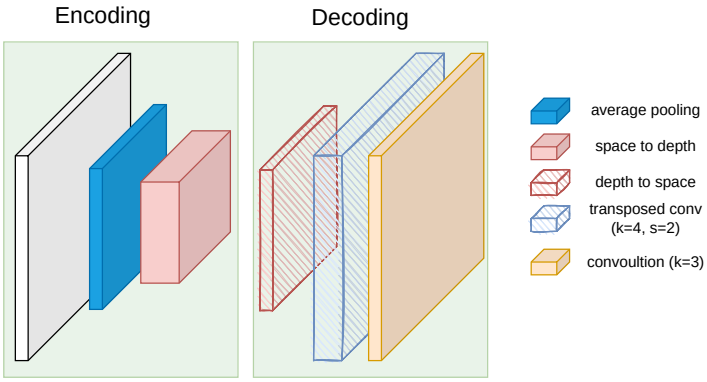


Figure 2: The autoencoder component of the model. This component is trained separately on probability maps, and the weights are frozen when incorporated into the main model. The downscaling component uses only pooling and space to depth operations to preserve probability values within the latent representation.

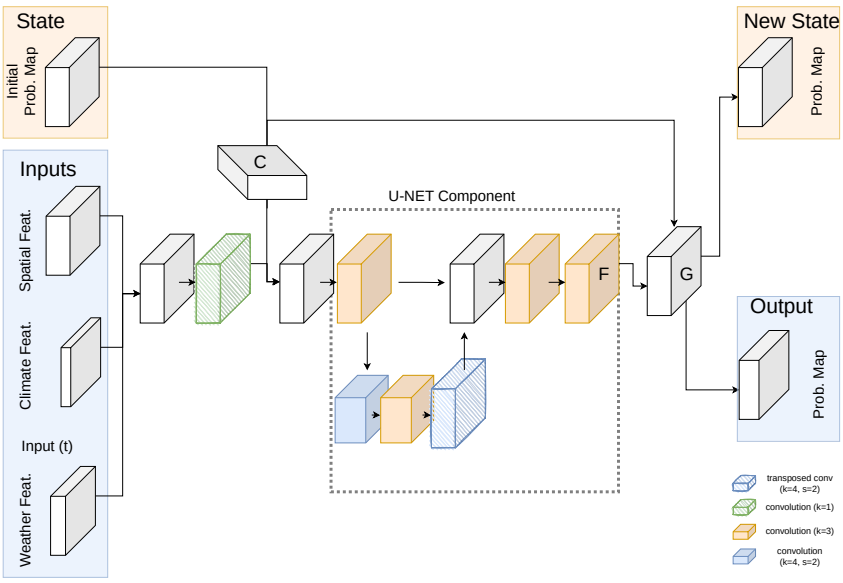
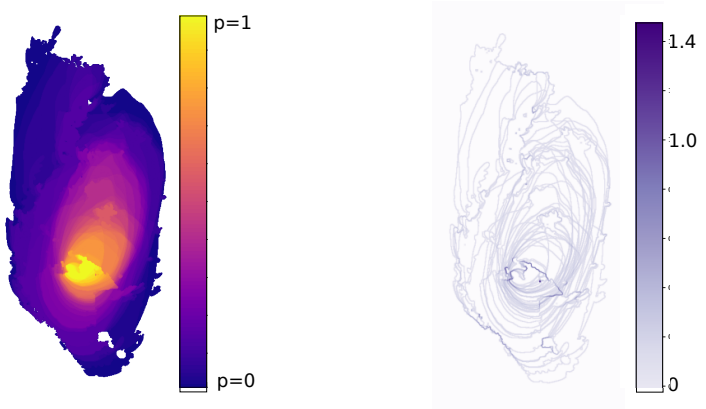


Figure 3: The inner component of the model. The latent spatial, climate, and weather time-series are concatenated. The latent arrival state is concatenated with this stack and is passed through a shallow U-Net [38]. The resulting output is used as a residual component. This residual is added to the latent arrival state, which is then used in the next time step. When the weather inputs are exhausted, the final latent arrival state is the output.



(a) (b)

Figure 4: **(a)**: Probability map (non-latent) of fire spread generated from an ensemble of 50 fire simulations. 14 hours have elapsed from the ignition time. The likelihood of a pixel being in the burned state is determined by the fraction of simulations in which a pixel is burned. Yellow indicates high likelihood, while blue indicates low likelihood. White indicates no burning in any simulation. Individual fire fronts within the ensemble lie on contours of the probability map. **(b)**: The magnitude of Sobel edges for the corresponding probability map. Gradients between probability contours correspond to individual fire fronts within the ensemble; these can be used as sources of fire growth.

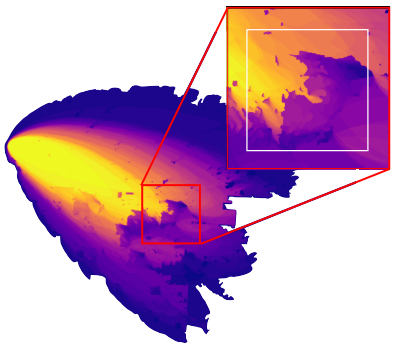


Figure 5: An example of a cropped image suitable for training. The original image is 1792×2048. The cropped region is 256×256 shown by the red box. The area outside the white box indicates the 32 pixels used for padding.

4 Empirical Evaluation

4.1 Data Preparation

We use a dataset comprised of 180 fire probability maps, split into 145 training and 35 validation samples. Likelihood estimates are given in half hour intervals throughout the duration of the sample. To generate each probability map, we produce an ensemble of 50 fire perimeter simulations using the SPARK platform [33]. For each time interval, the fire perimeters are calculated and the ensemble is flattened into a probability map. See Fig. 4 for an example.

The location of the fires is representative of regional South Australia. Realistic land classification maps[1] and topology data[2] are used. There is a large bias in land class distribution towards temperate grassland (55.4%), sparse shrubland (11.3%), water/unburnable (10.6%) and mallee heath (10.1%). Other classes represent 12.6% of the total area. Weather conditions[3] are sampled from days where the *forest fire danger index* is severe [14, 32]; the temperatures are typically high and the relative humidity is low.

The fires within an ensemble are identical, except that each time series of weather values is perturbed using the following noise model. For proof-of-concept we use a straightforward random walk noise model. At each time step (30 min interval), $\pm 3\%$ of the mean variable value is added. We choose 3% based on preliminary analysis; this value is large enough so that simulations within an ensemble display distinctly different features, while being small enough that the general shape of each fire is similar. Let x_i be the noise added at time i, V_j the noiseless variable from the time-series V at time j. We then define $\widetilde{V}_j = V_j + \sum_{i=1}^{j} x_i$ as a noisy weather source, where $x_i = \text{RAND}(\{-1, 1\}) \times 0.03 \times \overline{V}$, and $\overline{V}$ is the mean value of V. Each simulation within an ensemble uses an independently calculated $\widetilde{V}$.

The weather variables form the time series and are wind speed, wind direction, temperature and relative humidity. Due to use of degrees for wind direction within the weather data, the wind direction was perturbed by $\pm 2°$ at each step, rather than 3% of the mean value[4].

There are $N = 50$ unique time series for each variable (eg. temperature) of length T. Rather than designing the model to use N inputs at each time step for each variable, we use their mean and standard deviations, taken across the ensemble at a given time. This produces an input time series of size $(2, |V|, T)$, where $|V| = 4$ is the number of weather variables

Several pre-processing steps are applied to the features before they are presented to the model. The sample bounds, coordinate reference system, and resolutions (30m) of all spatial data are all made identical. Height maps are converted into gradient maps. The wind components of the weather data are converted from wind speed and direction to x and y components. Normalisation of weather and forcing terms is performed so that values fall within the range -1 to $+1$.

[1] Land classification datasets derived from Department of Agriculture and Water Resources *Land Use of Australia 2010-11* dataset.

[2] Topography data sets derived from Geoscience Australia SRTM-derived 1 Second Digital Elevation Models Version 1.0.

[3] Meteorological time series data sets derived from Australian Bureau of Meteorology automated weather station data.

[4] Percentage of the mean value cannot be used here, due to the nature of the measurement used for wind direction (degrees). If the mean value was used, winds with mean direction of $0°$ would never be perturbed, while winds with mean direction of $359°$ would have a large perturbation.

4.2 Training

The model training process performs further modification to the data, as follows. When a sample is selected, a random time within the burn duration is chosen as the initial probability map. The final probability map is taken D intervals later, and the corresponding weather variables are extracted.

Once a starting time is selected, a random square of the probability map is cropped, such that the central pixel j has a value, $x_j \in (0, 1)$, ie., excluding 0 and 1. We center the cropping on this pixel so that the cropped region is likely to contain pixels that have an intermediate probability of being burned. Such regions are likely to undergo dynamic changes within the time interval D used for training. We have empirically observed that in regions where pixel values are dominated by 0 or 1, the values are less likely to change during the time interval D. Fig. 5 shows an example of a cropped initial probability map suitable for training.

Each sample then undergoes a random rotation and/or reflection operation. This, as well as randomly choosing a starting time and a cropping location, is used for data augmentation with the aim of preventing over-fitting. The reduction in size reduces memory requirements, while the standardisation of sizes allows training to be batched.

The autoencoder component is trained using mean squared error (MSE) loss. For training the model, a loss function is used that evaluates how well the model reduces the MSE between predicted (y_p) and target (y_t) fire shapes compared to the MSE loss between target and initial (y_0) fire shapes. Let $\mathrm{MSE}(a, b)$ be the pixelwise mean squared error between images a and b. The loss $\mathcal{L}$ of fire state $\mathcal{P}$ is defined as $\mathcal{L}(\mathcal{P}) = \log_{10}\left(\left\{\mathrm{MSE}(y_p, y_t) + \tau\right\} / \left\{\mathrm{MSE}(y_0, y_t) + \tau\right\}\right)$. The term $\tau = 10^{-12}$ is used to remove singularities that arise if either MSE value approaches zero.

Models were implemented using TensorFlow [1], with the *Adam* optimiser [28] and a batch size of 16. The autoencoder was trained for 25 epochs and returned an MSE loss of 2.9×10^{-4}. The model was trained for 50 epochs. Unless explicitly stated otherwise, all layer activation functions in the model are Leaky Rectified Linear Units [10, 27].

4.3 Evaluation

We investigate training the model using $D = 1$ (30 min) and $D = 4$ (2 hours) difference between initial and final probability maps. In the former case, individual fires within the probability map may not significantly diverge, and the behaviour may be difficult for the model to learn. In the latter case, the dynamics are allowed more time to develop, but this comes at the cost of more invocations of the inner model. This recursive behaviour may lead to issues regarding vanishing/exploding gradients [21]; we found that the model weights failed to converge when using $D = 8$.

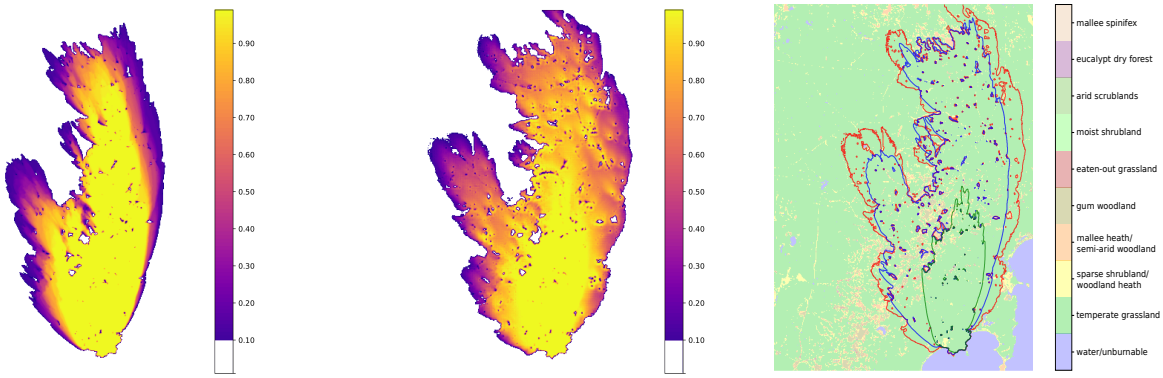
We also evaluate two configurations (A and B) of the inner model (shown in Fig. 3). In configuration A, layer C acts as an identity operation, simply passing through the latent probability map. In configuration B, layer C applies a Sobel edge filter with kernel size 3. We also train the model under various crop window sizes c. The amount of padding p is increased in proportion with the window size. For example a cropped square of 256 pixels has a padding size of 32 pixels, and a cropped square of 512 pixels has a padding size of 64 pixels.

We use two metrics to evaluate the performance: weighted MSE (WMSE) and Jaccard index. The WMSE is the average of MSE weighted by the amount of area that is burned in each sample. The area of land likely to be burned between initial and final states is given by $A(y_t^k, y_0^k) = \sum y_t^k - y_0^k$ for a sample k. We then define $\mathrm{WMSE}(y_t, y_p, y_0) = 1/(\sum_k A(y_t^k, y_0^k)) \sum_k \mathrm{MSE}(y_t^k, y_p^k) \times A(y_t^k, y_0^k)$, where we sum over all k samples. Lower values of WMSE indicate better performance.

The Jaccard index (aka Jaccard similarity coefficient) is defined as the intersection between two sets divided by their union [15]. We use the Jaccard index to gauge how similar the probability maps directly generated by the proposed model are to the probability maps indirectly generated via SPARK [33] (ie., where an ensemble of simulations is flattened into a probability map), with SPARK derived data treated as the ground-truth. Higher values of the Jaccard index indicate higher similarity. However, direct use of the Jaccard index on probability maps can be problematic, as there is no implicitly defined boundary on which to evaluate the index. As such, we use a threshold of 10% to define the boundary of the target and prediction sets (ie., probabilities less than 10% are ignored). This was chosen in order to evaluate the model's ability to capture the behaviour of reasonably likely events, while not penalising the model for missing highly unexpected behaviour. Furthermore, the 10% threshold may meet the needs of fire risk management (eg., decision planning and responsiveness), where low likelihood events are less likely to be considered as actionable.

Results presented in Table 1 show that $D = 4$ leads to better performance than $D = 1$. Furthermore, we observe a general reduction in the WMSE as the size of the cropped region is increased from 256 to 512 pixels. In terms of the Jaccard index, there is a notable improvement with configuration B compared to configuration A, which indicates that fire front detection via Sobel edge detection is a useful feature for determining ensemble fire dynamics.

Fig. 6 provides a graphical example of the performance on a sample fire over a long duration (14 hours). This example was produced using configuration B and a cropping size of 512 pixels during model training. Likelihoods below the 10% threshold have been truncated and are not shown.



(a)
Probability map from ensemble of simulations

(b)
Emulated probability map

(c)
Initial perimeter and final contours

Figure 6: **(a)**: A probability map that has been generated using an ensemble of 50 fires. The initial perimeter is marked as a green contour in panel (c). From this starting location the fire burns for a duration of 14 hours. Each fire within the ensemble is generated using weather inputs that are subjected to random walk noise. The ensemble is then flattened to generate the probability map. Values less than 10% are ignored. **(b)**: The corresponding probability map generated by the proposed model, with values less than 10% ignored. The model uses summary weather variables (mean and standard deviation) to reproduce/emulate the ensemble shown in panel (a). While the proposed model tends to overestimate high likelihood regions and underestimate low likelihood regions, the two maps qualitatively show a general agreement on shape. **(c)**: Contour lines for the probability maps at the 10% threshold. Blue colour represents the contour in panel (a), while red colour represents the contour in panel (b).

4.4 Comparison

As the proposed model is an evolution of the model presented in [6], in this section we evaluate the quality of probability maps directly generated by the proposed model (as per Section 4.3) against the quality of probability maps indirectly generated from ensembles of emulated fires produced via the model in [6]. In the latter case, the model is trained using noiseless weather variables on the same training dataset as the proposed model.

The quality is evaluated in terms of the Jaccard index (with contours defined at the 10% likelihood level), where we measure how similar the probability maps are to those indirectly generated via SPARK [33], which are treated as ground-truth. We also measure the time taken (in seconds) for each model to generate the probability maps, running on the same hardware (a modern computer with a multicore CPU and high-end GPU) and using the same software stack as per Section 4.2. For the direct method we choose the configuration with the lowest training and evaluation loss, which corresponds to setting shown in the last row of Table 1. For the indirect method, we vary the ensemble size for generating the probability maps; three ensemble sizes are evaluated: 10, 20 and 50.

The results shown in Table 2 indicate that for an ensemble size of 50 (the same size as used for training the direct approach), the indirect approach (ensembles of emulated fires produced via the model in [6]) produces somewhat higher quality maps. However, this comes at the cost of requiring considerably longer execution time, where the indirect approach is about 19 times slower than the direct approach. For an ensemble size of 20, the indirect approach produces probability maps which are roughly comparable in quality to the direct approach, but is about 7 times slower. For an ensemble size of 10, the indirect approach produces considerably lower quality maps and is still noticeably slower.

Overall, for the indirect approach there is a correlation between execution time and the quality of the resultant probability maps. However, increasing the size of the ensemble from 20 to 50 only leads to a minor increase in quality, at the cost of a large increase in the execution time. The considerably lower execution time of the proposed direct approach (while still producing reasonable quality probability maps) allows for the evaluation of more fire progression scenarios within a given time budget. This in turn can be useful for fire risk management, where large ensembles are typically used to explore large variable spaces.

Table 1: Results in terms of WMSE and Jaccard index for various settings and configurations. Each setting comprises c and p components, where c is size of the cropping window and p is the amount of padded pixels. D indicates the difference in time between initial and target probability maps used during training, measured in 30 minute intervals. Jaccard indices are calculated over the validation set for contours defined at the 10% likelihood level. The indices indicate how similar the probability maps directly generated by the proposed model are to the probability maps indirectly generated via SPARK [33].

Setting	Training WMSE	Validation WMSE	Jaccard
Config. A (no edges), D=1			
256c, 32p	0.0196	0.0211	0.528
384c, 48p	0.0172	0.0187	0.532
512c, 64p	0.0168	0.0173	0.561
Config. A (no edges), D=4			
256c, 32p	0.0145	0.0148	0.520
384c, 48p	0.0138	0.0151	0.563
512c, 64p	0.0136	0.0142	0.575
Config. B (edges), D=4			
256c, 32p	0.0129	0.0130	0.643
384c, 48p	0.0101	0.0110	0.679
512c, 64p	0.0087	0.0097	0.674

Table 2: Comparison of neural-based methods for estimating fire probability maps in terms of the Jaccard index (with contours defined at the 10% likelihood level) on the validation set, using SPARK [33] derived maps as ground-truth. Two methods are evaluated: ensembles of emulated fires produced via the model in [6], and the proposed model which directly generates probability maps. The proposed model uses the same setting as shown in the last row of Table 1. Execution time is reported in two forms: number of seconds, and relative to the execution time of the direct approach.

Method	Ensemble Size	Jaccard	time (sec.)	time (rel.)
ensembles of emulated fires	10	0.468	8.6	3.7
	20	0.707	17.6	7.6
	50	0.733	44.0	19.0
direct probability map	–	0.674	2.31	1.0

5 Concluding Remarks

In this paper we have demonstrated that deep neural networks can be constructed to directly generate probability maps for estimating wildfire spread. This direct approach can be used instead of indirectly obtaining probability maps via ensembles of simulations, which are typically computationally expensive. In comparison to ensembles of simulated fires obtained via the traditional SPARK platform [33], the proposed approach achieves reasonable quality probability maps when weather forecasts are exposed to random walk noise. On an evaluation set of 35 fires, the overall Jaccard index (similarity score) is 67.4%. Qualitatively, the proposed approach reproduces the correct general response to fuel class and wind direction, and the shape of the probability maps is comparable to those produced by traditional simulation.

The computational load for the proposed approach is concentrated in the model training process, rather than during deployment. When compared to a related neural model (emulator) which was employed to generate probability maps via ensembles of emulated fires, the proposed model produces competitive Jaccard similarity scores while being considerably less computationally demanding.

We observe that the model does not generate a perfectly smooth transition of probabilities from high to low, as observed in probability maps from ensembles of simulated fires. This will likely limit the capability for the model to predict accurately over long durations. Further refinement of the architecture, model restraints, summary variables and incorporation of variable distributions may yield better results.

A further area of research would be the incorporation of better noise models for weather variables. In this work we have used a straightforward noise model, aimed for demonstrating proof-of-concept. This noise model was not developed to be representative of the complexities in weather forecasting. A more detailed discussion of more comprehensive weather models that incorporate uncertainty can be found in [34, 35, 43]. In our noise model, any two fires within an ensemble are likely to smoothly diverge from each other over time, except in cases where natural barriers cause bifurcating behaviour. In more realistic settings, bifurcation may occur due to rapid changes in wind direction that are predicted to occur at various times within an ensemble of simulations. It may be possible to capture the behaviour of more complex noise models by using binning methods to generate summary variables, rather than simply using mean and standard deviation.

More sophisticated fire spread models (for example ones that incorporate ember attacks [41, 47]) may require very large ensembles in order to reliably gauge the behaviour of fires. Generating likelihood estimates through very large ensembles is likely to be computationally taxing. Direct generation of probability maps, as proposed in this work, may be very beneficial in these contexts. Finally, we note that the proposed approach is not specific to wildfire prediction, and other similar spatio-temporal problems may benefit from direct likelihood estimation, such as diffusion of pollution, pest and disease spread, and flood impact models.

Acknowledgements. We would like to thank our colleague Peter Wang for providing results from the SPARK simulation platform.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, et al. TensorFlow: A system for large-scale machine learning. In *USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

[2] F. Abid. A survey of machine learning algorithms based forest fires prediction and detection systems. *Fire Technology*, 57(2):559–590, 2021.

[3] R. Alizadeh, J. K. Allen, and F. Mistree. Managing computational complexity using surrogate models: a critical review. *Research in Engineering Design*, 31(3):275–298, 2020.

[4] F. Allaire, V. Mallet, and J.-B. Filippi. Emulation of wildland fire spread simulation using deep learning. *Neural Networks*, 141:184–198, 2021.

[5] L. S. Bastos and A. O'Hagan. Diagnostics for Gaussian process emulators. *Technometrics*, 51(4):425–438, 2012.

[6] A. Bolt, C. Huston, P. Kuhnert, J. J. Dabrowski, J. Hilton, and C. Sanderson. A spatio-temporal neural network forecasting approach for emulation of firefront models. In *Signal Processing: Algorithms, Architectures, Arrangements, and Applications*, 2022.

[7] K. Bot and J. G. Borges. A systematic review of applications of machine learning techniques for wildfire management decision support. *Inventions*, 7(1):15, 2022.

[8] D. M. Bowman, C. A. Kolden, J. T. Abatzoglou, F. H. Johnston, G. R. van der Werf, and M. Flannigan. Vegetation fires in the anthropocene. *Nature Reviews Earth & Environment*, 1(10):500–515, 2020.

[9] J. Burge, M. Bonanni, M. Ihme, and L. Hu. Convolutional LSTM neural networks for modeling wildland fire dynamics. arXiv:2012.06679, 2021.

[10] F. Chollet et al. Keras, 2015. https://github.com/fchollet/keras.

[11] S. Conti, J. P. Gosling, J. E. Oakley, and A. O'Hagan. Guassian process emulation of dynamic computer codes. *Biometrika*, 96(3):663–676, 2009.

[12] J. J. Dabrowski, D. E. Pagendam, J. Hilton, C. Sanderson, D. MacKinlay, C. Huston, A. Bolt, and P. Kuhnert. Bayesian physics informed neural networks for data assimilation and spatio-temporal modelling of wildfires. *Spatial Statistics*, 55:100746, 2023.

[13] P.-E. Danielsson and O. Seger. Generalized and separable Sobel operators. In *Machine Vision for Three-Dimensional Scenes*. Academic Press, 1990.

[14] A. J. Dowdy, G. A. Mills, K. Finkele, and W. de Groot. Australian fire weather as represented by the McArthur Forest Fire Danger Index and the Canadian Forest Fire Weather Index. CAWCR Technical Report No. 10, 2009.

[15] T. J. Duff, D. M. Chong, and K. G. Tolhurst. Indices for the evaluation of wildfire spread simulations using contemporaneous predictions and observations of burnt area. *Environmental Modelling & Software*, 83:276–285, 2016.

[16] R. Egele, R. Maulik, K. Raghavan, B. Lusch, I. Guyon, and P. Balaprakash. AutoDEUQ: Automated deep ensemble with uncertainty quantification. In *International Conference on Pattern Recognition*, pages 1908–1914, 2022.

[17] M. A. Finney. FARSITE: Fire area simulator–model development and evaluation. Research Paper RMRS-RP-4, U.S. Department of Agriculture, 1998.

[18] D. Gladish, D. Pagendam, L. Peeters, P. Kuhnert, and J. Vaze. Emulation engines: choice and quantification of uncertainty for complex hydrological models. *Journal of Agricultural, Biological, and Environmental Statistics*, 23(1):39–62, 2018.

[19] T. Gneiting and A. Raftery. Weather forecasting with ensemble models. *Science*, 310:248–249, 2005.

[20] N. Gunawardena, G. Palotta, M. Simpson, and D. D. Lucas. Machine learning emulation of spatial deposition from a multi-physics ensemble of weather and atmospheric transport models. *Atmosphere*, 12(8):953, 2021.

[21] B. Hanin. Which neural net architectures give rise to exploding and vanishing gradients? *Advances in Neural Information Processing Systems*, 31, 2018.

[22] J. L. Hodges and B. Y. Lattimer. Wildland fire spread modeling using convolutional neural networks. *Fire Technology*, 55(6):2115–2142, 2019.

[23] A. A. Ismail, H. C. Bravo, and S. Feizi. Improving deep learning interpretability by saliency guided training. In *Advances in Neural Information Processing Systems*, volume 34, pages 26726–26739, 2021.

[24] P. Jain, S. C. Coogan, S. G. Subramanian, M. Crowley, S. Taylor, and M. D. Flannigan. A review of machine learning applications in wildfire science and management. *Environmental Reviews*, 28(4):478–505, 2020.

[25] K. Kashinath, M. Mustafa, A. Albert, J.-L. Wu, C. Jiang, S. Esmaeilzadeh, K. Azizzadenesheli, R. Wang, A. Chattopadhyay, et al. Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200093, 2021.

[26] M. F. Kasim, D. Watson-Parris, L. Deaconu, S. Oliver, P. Hatfield, D. H. Froula, G. Gregori, M. Jarvis, S. Khatiwala, et al. Building high accuracy emulators for scientific simulations with deep neural architecture search. *Machine Learning: Science and Technology*, 3(1):015013, 2021.

[27] M. Khalid, J. Baber, M. K. Kasi, M. Bakhtyar, V. Devi, and N. Sheikh. Empirical evaluation of activation functions in deep convolutional neural network for facial expression recognition. In *International Conference on Telecommunications and Signal Processing*, 2020.

[28] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[29] W. B. Leeds, C. K. Wikle, J. Fiechter, J. Brown, and R. F. Milliff. Modeling 3-D spatio-temporal biogeochemical processes with a forest of 1-D statistical emulators. *Environmetrics*, 24(1):1–12, 2013.

[30] M. Leutbecher and T. N. Palmer. Ensemble forecasting. *Journal of Computational Physics*, 227:3517–3539, 2007.

[31] V. Mallet, D. Keyes, and F. Fendell. Modeling wildland fire propagation with level set methods. *Computers & Mathematics with Applications*, 57(7):1089–1101, 2009.

[32] A. G. McArthur. Fire behaviour in eucalypt forests. Leaflet No. 107, Forestry and Timber Bureau, Department of National Development, Australia, 1967.

[33] C. Miller, J. Hilton, A. Sullivan, and M. Prakash. SPARK – a bushfire spread prediction tool. In *Environmental Software Systems, Infrastructures, Services and Applications, IFIP Advances in Information and Communication Technology, vol. 448*, pages 262–271, 2015.

[34] A. Murphy. *Probability, Statistics, and Decision Making in the Atmospheric Sciences*. CRC Press, 1985.

[35] T. N. Palmer. Predicting uncertainty in forecasts of weather and climate. *Reports on Progress in Physics*, 63:71, 2000.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[37] D. Radke, A. Hessler, and D. Ellsworth. FireCast: Leveraging deep learning to predict wildfire spread. In *International Joint Conference on Artificial Intelligence*, pages 4575–4581, 2019.

[38] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. Lecture Notes in Computer Science, vol. 9351, pp. 234–241, 2015.

[39] C. Sanderson, D. Pagendam, B. Power, F. Bennett, and R. Darnell. Opportunistic emulation of computationally expensive simulations via deep learning. In *International Congress on Modelling and Simulation*, pages 673–679, 2021.

[40] A. Scott, D. Bowman, W. Bond, S. Pyne, and M. Alexander. *Fire on Earth: An Introduction*. Wiley, 2013.

[41] A. Sharifian and J. Hashempour. A novel ember shower simulator for assessing performance of low porosity screens at high wind speeds against firebrand attacks. *Journal of Fire Sciences*, 34(4):335–355, 2016.

[42] M. Sit, B. Z. Demiray, Z. Xiang, G. J. Ewing, Y. Sermet, and I. Demir. A comprehensive review of deep learning applications in hydrology and water resources. *Water Science & Technology*, 82(12):2635–2670, 2020.

[43] J. Slingo and T. Palmer. Uncertainty in weather and climate prediction. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369(1956):4751–4767, 2011.

[44] S. Sung, Y. Li, and L. Ortolano. WildfireNet: Predicting wildfire profiles (student abstract). *AAAI Conference on Artificial Intelligence*, 35(18):15905–15906, 2021.

[45] K. Tolhurst, B. Shields, and D. Chong. Phoenix: development and application of a bushfire risk management tool. *Australian Journal of Emergency Management*, 23(4):47–54, 2008.

[46] C. Wang, Z. Zhao, Q. Ren, Y. Xu, and Y. Yu. Dense U-net based on patch-based learning for retinal vessel segmentation. *Entropy*, 21:168, 2019.

[47] H. Wang. Ember attack: Its role in the destruction of houses during ACT bushfire in 2003. In *10th Biennial Australasian Bushfire Conference, Brisbane, Australia*, 2006.

[48] W. Wu, R. Emeron, Q. Duan, A. W. Wood, F. Wetterhall, and D. E. Robertson. Ensemble flood forecasting: Current status and future opportunities. *WIREs Water*, 7(3), 2020.